UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/769,535	01/30/2004	Milton E. Moskowitz	H0005134- -1623	8642

128 7590 11/26/2008

HONEYWELL INTERNATIONAL INC.
101 COLUMBIA ROAD
P O BOX 2245
MORRISTOWN, NJ 07962-2245

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
11/26/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/769,535	MOSKOWITZ ET AL.	
	Examiner	Art Unit	
	TUAN A. VU	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *17 September 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-18,21 and 22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-18, 21,22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 8/17/08.

As indicated in Applicant's response, claims 1-18, 21-22 have been amended, and claims 19-20 canceled. Claims 1-18, 21-22 are pending in the office action.

Claim Objections

2. Claims 1-3, 7, 9, 12, 16 are objected to because of the following informalities:

Claims 2-3 are objected to because of typographical error in repeated terms: that is, "*if the if the* generated code" (cl. 2 li. 6-7) and "if *the the* generated code" (cl. 3 6-7);

Claim 1, 7, 9, 12, 16 are objected to because of verb accordance impropriety; that is, "one of more lines ... *does not* include" (cl. 1, li. 15; cl. 7, li. 18; cl. 12, li. 15) and "the plurality of lines *includes*" (cl. 9, li. 4; cl. 16, li. 3).

Appropriate correction is required.

Claim Rejections - 35 USC § 112

3. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

4. Claims 5, 11, 15-18 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

As per claims 5, 11, 18, the 'comparing' limitation as to matching 'a first header information section in the generated code' with 'a second header information section in the expected code' is not found as having a clear and proper description in the Specifications. Scanning the Disclosure for 'header section' it is observed that Figure 3B pg. 4 depicts header section (or header file) of the generated code *Ccrdemo.c*, and paragraph 0032, pg. 9 describes how shorthand notation labels (<S4>) of a model are used as indicators based on which to be place generated code construct of the header file. Shorthand labels are syntax in a special form but cannot be construed as 'header information section' of a expected code. Nowhere in the Specifications is the 'expected code' is not disclosed as explicitly including a main body and an header section portion; and as described this type of code amounts to mere meta-information (see line 342, Fig. 3B(Continued); para 0038, 0044, pg. 11) implemented as syntactic form using bracket and labels < > like tagged language. Thus, tagged format cannot be equated to 'a second header information section' of the expected code, because this 'expected code' is not disclosed as comprising sections one of which is 'header section' as claimed in the 'comparing' step. The 'second header section' is not given any weight because it is deemed that the inventor does not possess this feature at the time the invention was made. The above 'comparing' step will be treated as though 'first header section' in the generated code is matched against expected code that is derived from meta-information from the model.

Claims 15-17 recite 'comparing each line in the expected code **to determine** if the plurality of lines ...', in the sense that the plurality of lines of generated code are in expected form(cl. 15) or include correct number of lines (cl. 16), or in proper logical order (cl. 17) solely based on a *comparing* step. However, the language regarding this 'comparing' teaches

comparing each of the lines of the expected code, no mention about whether each line is compared to the generated code. The proper interpretation is that each and all lines of the expected are compared to one another. The disclosure teaches a syntactic tagged format (with enclosed labels) based on which to model code construction, or dictate commands therefor; e.g. matching each of such tagged labels with constructs of the generated code. But nowhere is there explicit description about just comparing each or every line of such special tagged format (i.e. expected code) **to determine** correctness as set forth above. Based on what is clearly disclosed (para 0038, 0044, pg. 11) it is observed that matching of labels sequenced in (**belonging to one line of**) the tagged special command/format (i.e. expected code) with existing constructs of the generated code cannot be construed as comparing *each line of expected code*. The 'comparing each line in the expected code' will be given no weight because it is deemed that the inventor does not possess this limitation at the time the invention was made in order to achieve the *determining* as set forth above. The limitation is treated as though the determining is based on expected syntax derived from a model.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1-18, 21-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Charisius et al, USPN: 6,983,446 (hereinafter Charisius)

As per claim 1, Charisius discloses a method for verifying computer code having a plurality of lines generated by a code generating module from a model file of a system including a plurality of functions generated by a model module, the method comprising:

processing the model file (TMM – Fig. 2; Fig. 3; Fig. 13-14), by a verification module (e.g. Table 1 → table 18, col. 10 to col. 20; Fig. 19B, 20A – Note: Processing TMM model then invoke a verification tool reads on verification based on processing code from a model), to determine values, inputs, outputs, function type, and syntax for each of the plurality of functions (e.g. Fig. 21-23; QA audit- Fig. 19B; number of attributes, of import -Table 1, col. 10; Table 3, col. 11; classes … formal parameters, return types, local variables -Table 4 col. 11; Table 11, col. 15; Table 13 col. 17);

determining expected code for the generated computer code (Steps 2000-2004, 2008, Fig. 20A, Table 1 –> Table 18, col. 10-20 – Note: Processing TMM model then invoke a verification tool using auditing criteria on naming, declaration, content cohesion, encapsulation coding style **reads on** processing to determine code compliancy, and metrics based on which to enforce language compliancy in terms of programming semantic and syntactic errors or audit violation – *Basic Metrics, Cohesion Metrics, Complexity Metrics, inheritance Metrics, maximum metrics, Ratio Metrics, Style audits, Declaration audits, Name style, Superfluous content*; that is, expected code having syntax or form being language compliant) based on the determined values, inputs, outputs, function type, and syntax for the generated computer code (e.g. number of attributes, of import -Table 1, col. 10; Table 3, col. 11; classes … formal parameters, return types, local variables -Table 4 col. 11; Table 11, col. 15; Table 13 col. 17; col. 22 lines 40-55);

comparing each line of the generated computer code and the expected code to determine if the generated computer code includes correct values, correct inputs, correct outputs, correct functions, and correct syntax (e.g. Fig. 4; Fig. 19B-C; Fig. 8B-C; code 1302 – Fig. 13 – Note: actual source code using the SCI reads on *generated code* lines 810, 812, 1302; and audit module to validate expected code being model specifications that encompass rules definitions – audited by metrics - reads on comparing expected and lines of generated; Fig. 3; Fig. 13-14; col. 5, lines 61-64; col. 6, lines 12-22; Note: *definitions, templates*, Fig. 8B-C; col. 7, lines 63 to col. 8, line 21; Fig. 19A, 20A – reads on expected form and syntax being determined in TMM form via template of lines – col. 16, lines 9-14 – and/or metrics as correct values, input/outputs, functions syntax – see Table 1, Table 3, Table 4 from above);

transmitting an error message when audit is performed if lines of the generated computer code does not include a correct syntax based on the comparison (e.g. error message … conform to predefined styles; error message when an audit is performed – col. 18 lines 50-55; col. 19 lines 28-34, 52-55; error message - col. 20 lines 25-67).

But Charisius does not explicitly disclose transmitting the error message *if one or more lines of generated code does not include a correct value, correct input, correct output, a correct function*. Using a QA module operating on predefined audit error types as set forth above (Fig. 3; Fig. 13-14; col. 5, lines 61-64; col. 6, lines 12-22; *definitions, templates*, Fig. 8B-C; col. 7, lines 63 to col. 8, line 21; Fig. 19A, 20A –col. 16, lines 9-14 – and/or metrics as correct values, input/outputs, functions syntax – see Table 1, Table 3, Table 4 from above), Charisius discloses error message to be shown to developers regarding many respects, such as syntax, format parameters of a function, declaring of class or members and this is indicative that any violation

determined by a verification or audit process to input/output declarations, or correct values or functions would be raised by the validation process. Based on Charisius's and well known practice (in software development) of using a verification/auditing process to systematically notify the developers using visual means on any propriety issue regarding code form or implementation thereof, it would have been obvious for one skill in the art at the time the invention was made to implement the error messages by the source code auditing in Charisius so that error message would be displayed when auditing a line of expected source if the line does not include specifically a correct value, correct input, correct output, a correct function. One of ordinary skill in the art would be motivated to do so because not having such proper value or function, correct input or output thereof a code might not properly compile notably when such type of deficiencies would necessarily fall under the types or style of errors contemplated by a auditing tool based on the above.

As per claim 2, Charisius teaches comparing each line of the generated computer code and the expected code to determine if the generated computer code is missing a line of code (is missing – col 15 lines 47-48; col 25 lines 55-59); and, by virtue of the rationale in claim 1, transmitting the error message if the if the generated computer code is missing the line of code.

As per claim 3, Charisius teaches comparing each line of the generated computer code and the expected code to determine if the generated computer code includes any an extraneous line of code (col. 20 Table 18; col. 22 lines 61-67); and by virtue of the rationale in claim 1, transmitting the error message if the the generated computer code includes the extraneous line of code.

As per claim 4, Charisius discloses comparing each line of the generated computer code and the expected code to determine if the [the] generated computer code is in a logical order (ordered properly – col. 33, lines 55-60 ; col. 34 lines 10-44) and by virtue of the rationale in claim 1, transmitting the error message if the generated computer code is not in the logical order.

As per claim 5, Charisius discloses comparing a first header information section in the generated computer code and second header information section in the expected code to determine if the first header information section matches (*Declaration* – cols. 25-26; match a declaration, col. 37, lines 1-37- Note: generated source code or class package declaration with respect to expected declaration in OO class or Use case package – see Fig. 14-15, 22 -- in an *audit* instance reads on comparing header of a class signature declaration – Note: second header information treated as any form of expected header – see USC 112, 1st para) the second header information section; and (re rationale in claim 1) transmitting the error message if the first header information section does not match the second header information section.

As per claim 6, Charisius discloses comparing a first declared variable section in the generated computer code and a second declared variable section in the expected code (col. 29 lines 7 to col. 30 lines 67; col. 25 lines 20-52 – Note: fixed source code based on positions of declarations, declared type of member with respect to their signature and constructor with respect to a parent class reads on comparing generated variable section with variable section of expected code) to determine if the first declared variable section matches the second declared variable section; and (refer to rationale in claim 1) transmitting the error message if the first declared variable section does not match the second declared variable section.

As per claim 7, Charisius discloses computer-readable storage medium containing a set of instructions for verifying a generated computer code having a plurality of lines from a code generating module, the generated computer code automatically generated from a model file of a system having a plurality of functions and created by a model module, the set of instructions comprising:

code that reads in the model file (TMM – Fig. 2; Fig. 3; Fig. 13-14;Table 1 → table 18, col. 10 to col. 20; Fig. 19B, 20A – Note: Processing TMM model then invoke a verification tool reads on verification based on processing code from a model);

code that determines values, inputs, outputs, function type, and syntax for each of the plurality of functions in the generated computer code (refer to claim 1);

code that generates an expected computer code based on the determined values, inputs, outputs, function type, and syntax (refer to claim 1);

code that reads in the generated computer code (Note: comparing source code with expected reads on reads in the generated source code); code that compares each line in the generated computer code and the expected code to determine if the generated computer code includes the determined values, inputs, outputs, function type, and syntax in the expected computer code (refer to claim 1); and

code that transmits an error message if one or more lines in the generated computer code does not include determined syntax based on the comparison (refer to claim 1)

But Charisius does not explicitly disclose error message if the generated code does not include a determined value, a determined input, a determined output, a determined function based on the comparison. This limitation has been rendered obvious in claim 1.

As per claim 8, Charisius discloses code that compares each line in the generated computer code and the expected code (refer to claim 7); and by virtue of the rationale set forth in claim 1, teaches code that transmits the error message if one of the plurality of lines does not include the determined *value*, the determined *input*, the determined *output*, the determined *function*, the determined *syntax*, or combinations thereof.

As per claim 9, Charisius discloses code that compares each line in the plurality of lines and the expected code to determine if the plurality of lines includes a line of code that is not determined in the expected code (refer to claim 2); and (by virtue of the rationale in claim 1) discloses code that transmits the error message if the generated computer code includes any line of code not determined in the expected computer code.

As per claim 10, Charisius discloses wherein the set of instructions further comprises: code that compares each line in the plurality of lines and the expected code to determine if the plurality of lines are in a correct logical order (refer to claim 4) and (by virtue of the rationale in claim 1) discloses code that transmits the error message if the plurality of lines are not in the correct logical order.

As per claim 11, Charisius discloses wherein the set of instructions further comprises: code that compares a first header information section in the generated computer code and a second header information section in the expected code to determine if the first header information section matches the second header information section (refer to claim 5); and (by virtue of the rationale in claim 1) discloses code that transmits the error message if the first header information section does not match the second header information section.

As per claim 12, Charisius discloses a system for verifying the contents of a generated computer code having a plurality of lines generated by a code generating module from a model file including a plurality of functions generated by a model module, comprising:

a processor operable to:

process the model file (refer to claim 1) to determine values, inputs, outputs, function type, and syntax for each of the plurality of functions,

determine expected computer code for the generated computer code based on the determined values, inputs, outputs, functions type, and syntax for the generated computer code(refer to claim 1)

compare each line in the generated computer code with each corresponding line in the expected computer code to determine (refer to claim 1) if the generated computer code includes correct values, correct inputs, correct outputs, correct functions, and correct syntax, and

transmit an error message if each one or more lines in the generated computer does not include, or a correct syntax based on the comparison (refer to claim 1); and

a display configured to display the error message (Note: error message reads on display coupled to the processor) the display coupled to the processor.

But Charisius does not explicitly disclose transmitting the error message if one or more lines of generated code do not include a correct value, a correct input, a correct output, a correct function. This limitation has been addressed in claim 1.

As per claim 13, Charisius discloses wherein the error message indicates if a line is missing in the generated computer code (refer to claim 2).

As per claim 14, Charisius discloses wherein the error message indicates if a line of the generated computer code has any additional content (e.g. *True ... False literals should not be used ... amount of meaningless code ... use of type casts not necessary ... 'abstract' considered obsolete ... should not use parentheses* – col. 19 lines 47 to col. 20 line 23; col. 16 line 58 to col. 17, line 37).

As per claims 15-16, Charisius discloses wherein the processor is operable to compare each line in the expected code to determine if the plurality of lines are in an expected form (Fig. 4; Fig. 19B-C; Fig. 8B-C; code 1302 – Fig. 13; Fig. 3; Fig. 13-14; col. 5, lines 61-64; col. 6, lines 12-22; Note: *definitions, templates*, Fig. 8B-C; col. 7, lines 63 to col. 8, line 21; Fig. 19A, 20A Note: refer to USC 112, 1st paragraph regarding "compare each line in the expected code", i.e. comparing treated as determine language/information derived of a model to determine if such information/language or code is in expected form), and transmit the error message if one or more of the lines of code in the plurality of lines do not match the expected form (e.g. error message … conform to predefined styles; error message when an audit is performed – col. 18 lines 50-55; col. 19 lines 28-34, 52-55; error message - col. 20 lines 25-67);

wherein the processor is operable to compare each line in the expected code (refer to USC 112, 1st para rejection) to determine if the plurality of lines includes one or more lines of extraneous code (refer to rationale in claim 3).

As per claim 17, Charisius discloses wherein the processor is operable to compare each line in the expected code (Note: comparing treated as comparing model derived constructs with expected code – see USC 112 rejection) to determine if each line is in a logical order, and

transmit the error message if any line in the plurality of lines is not in logical order (refer to claim 4).

As per claim 18, Charisius discloses wherein the processor is operable to compare a first header information section in the generated computer code to a second header information section in the expected computer code to determine if the first header information section matches the second header information section (refer to claim5), and transmit (by virtue of the rationale in claim 1) the error message if the first header information section does not match the second header information section.

As per claim 21, with reference to claim 1, Charisius discloses steps of:

comparing each line in the generated computer code and the expected computer code to determine if the plurality of lines are complete (col. 16 line 55 to col. 17 line 37; avoid empty catch block … without empty body – Table 17 col. 18-19); and transmitting an error message if the plurality of lines are incomplete (refer to obviousness rationale of claim 1).

As per claim 22, with reference to claim 7, Charisius discloses wherein the set of instructions further comprises code that

compares each line in the generated computer code to the expected computer code to determine if the generated computer code is complete (refer to claim 21) and code that transmits the error message (refer to obviousness set forth in claim 1) if the generated computer code is incomplete.

Response to Arguments

7. Applicants arguments have been considered but they are not persuasive. Following are the Examiner's observations in regard thereto.

35 USC § 103 Rejection:

(A) Applicants have submitted that the process of graphically automatically upgrading a source code is not the same as having two codes being generated from a same model file, especially for verify correctness of values, input/outputs, function type, and syntax (Appl. Rmrks pg. 17 top para). There is no generation of the expected code anywhere in the language of claims 1, 7, 12; that is, expected code is treated as a expected code determined internally by the verification process when this process audits the syntax of the model based source code in Charisius. The language of the claim is deemed matched, while the limitations regarding correctness of values, input/outputs, function type have been addressed according to interpretation. Arguing based on any added limitations would be deemed moot because such argument would not be deemed commensurate with a previous Office Action.

(B) Applicants have submitted that the relied upon OSA does not cure the deficiencies of Charisius (Appl. Rmrks, pg. 17, bottom para). The mention about a combination with OSA regarding a added limitation such as 'correct values, correct inputs … and correct syntax' is deemed not commensurate with the previous Office Action, wherein the previous claim language do not include these amended limitations.

(C) Applicants have submitted that regarding (as in claims 7-11, 22) 'compares each line in the generated code' Charisius, in view of OSA does not teach or suggest 'to determine if the generated code … includes the determined values … function type and syntax' (Appl. Rmrks pg. 18, top half). The added limitations have been interpreted and addressed with an obviousness rationale which has been necessitated by the Amendments, and the mention about OSA is

deemed not commensurate with any actual Office Action (where no OSA reference was included).

(D) Applicants have submitted that regarding the 'comparing … to determine' limitation in claims 12-18, Charisius, in view of OSA, does not teach or suggest 'to determine if the generated code … includes the determined values … function type and syntax' (Appl. Rmrks pg. 18, bottom). The argument using OSA is deemed not matching any Office Action particularly when this is based on the added claim language. The argument stands not sufficient to overcome the current Office Action.

In all, the claims stand rejected as set forth in the Office Action, including grounds of rejection that are responsive to the current Amendments.

Conclusion

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

November 23, 2008